Correction of Uniformly Noisy Distributions to Improve Probabilistic Grammatical Inference Algorithms*

Amaury Habrard and **Marc Bernard** and **Marc Sebban**
EURISE – Université Jean Monnet de Saint-Etienne
23, rue du Dr Paul Michelon – 42023 Saint-Etienne cedex 2 – France
{amaury.habrard,marc.bernard,marc.sebban}@univ-st-etienne.fr

Abstract

In this paper, we aim at correcting distributions of noisy samples in order to improve the inference of probabilistic automata. Rather than definitively removing corrupted examples before the learning process, we propose a technique, based on statistical estimates and linear regression, for correcting the probabilistic prefix tree automaton (*PPTA*). It requires a human expertise to correct only a small sample of data, selected in order to estimate the noise level. This statistical information permits us to automatically correct the whole *PPTA* and then to infer better models from a generalization point of view. After a theoretical analysis of the noise impact, we present a large experimental study on several datasets.

Keywords. probabilistic grammatical inference, noisy data, distribution correcting

Introduction

Nowadays, machine learning algorithms have to process huge amount of data. Beyond this algorithmic constraint, they also have to deal with an important presence of noise, due to different reasons such as typing, transfer or experimental errors. To tackle these problems, a lot of data reduction techniques have been proposed to remove, either noisy examples (usually called *prototype selection* (Brodley & Friedl 1996)), or noisy features (called *feature selection* (John, Kohavi, & Pfleger 1994)). These approaches require generally not only positive but also negative instances of the concept to learn and use a discrimination criterion (entropy measures (Sebban & Nock 2000), classification success rate (Brodley & Friedl 1996), *etc*.) to detect the presence of noise. In this context, a noisy example is considered as a positive (resp. negative) instance which should be negatively (resp. positively) labeled in the absence of noise. Unfortunately, we often do not have access to negative examples. In natural language processing, for example, it is hard to enumerate a lot of counter-examples of the language. Thus, in this context, standard data reduction techniques are totally unsuitable when only positive examples are available.

*This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

Algorithms that learn only from positive data usually use statistical information in the learning sample to generate a model defining a probability distribution on positive data. Such approaches are usually called stochastic methods. In this context, a current trend is to consider that a noisy instance is a weakly probable example, not representative of the underlying distribution. Indeed, if such data (usually called *outliers*) would cover a significant part of the probability density function, they would become a relevant part of the concept to learn. We can then think that a weakly probable example is due to the presence of noise, or at least to a weak relevance for characterizing the target concept. Then, it would be reasonable to remove it, as it has already been done efficiently for tree-structured data in (Habrard, Bernard, & Sebban 2003b). However, as we mentioned before, stochastic models require the estimation of probabilities and then large learning sets. In such a context, we think that a distribution correction procedure is more suited than a data reduction technique aiming at definitively deleting noisy instances. Moreover, noise can not only generate *outliers*, but also modify the probability of some examples that characterize the concept to learn. These consequences can be dramatic for stochastic learning algorithms, such as probabilistic grammatical inference algorithms.

Grammatical inference (de la Higuera 2005) is a domain of machine learning whose goal is to learn language models from a set of sequences. Probabilistic grammatical inference aims at learning a probabilistic automaton which defines a distribution on the language recognized by this automaton. Learning data (only positive) are supposed to have been generated following a theoretical probability distribution. The goal is then to learn the target automaton which has generated the data. We focus here, more specifically, on algorithms based on state merging techniques (Carrasco & Oncina 1994; Thollard, Dupont, & de la Higuera 2000). They generally use the same base principle consisting in building, during a first step, a specific probabilistic automaton (called the *PPTA* for *Probabilistic Prefix Tree Acceptor*) recognizing only the learning examples. Then, for allowing generalization performances, the inference process is achieved by iteratively merging states considered as equivalent from a statistical point of view.

In this article, we focus on learning probabilistic automata from uniformly noisy data. Our objective is to correct this

noise and then to improve the behavior of standard inference algorithms on modern databases which often contain noisy data. For these databases, the theoretical target automaton is obviously unknown, that justifies the use of a statistical method to assess the level of noise. Few studies have been presented in this context, except (Sebban & Janodet 2003) in regular inference and (Habrard, Bernard, & Sebban 2003a) in probabilistic inference by removing noisy tree-structured data. We aim, here, at correcting the whole learning set distribution (more precisely a representation of it under the form of a *PPTA*), using a small noise-free sample. The existence of such a sample appears necessary, because it seems obviously difficult to detect noisy data only from positive examples. We think that this constraint is not too difficult to satisfy in domains where an expert evaluation is possible, such as in molecular biology or in natural language processing, for example. We show now how this corrected sample can be used to estimate the noise level in data and then to automatically correct the *PPTA*. The underlying principle of our estimation method, is based on a linear regression technique, looking for a correlation between frequencies in the *PPTA* built from the noisy sample and the one built from the manually corrected sample. Once the noise level is estimated, we correct noisy frequencies using confidence intervals which take into account the sizes of the samples.

This article is organized as follows. We begin with a brief recap on probabilistic automata and their learning algorithms. Then, we present a theoretical analysis on the impact of a uniformly distributed noise on learning data, before describing our approach to correct the data distribution. Finally, we detail our experimental validation before concluding.

Learning Probabilistic Finite State Automata

Probabilistic Finite State Automata (PFSA) are a probabilistic extension of Finite State Automata and define a probability distribution on strings.

Definition 1 *A PFSA A is a 6-tuple $(Q, \Sigma, \delta, p, q_0, F)$. Q is a finite set of states. Σ is the alphabet. $\delta : Q \times \Sigma \to Q$ is the transition function, δ' denotes its transitive closure. $p : Q \times \Sigma \to [0, 1]$ is the probability of a transition. q_0 is the initial state. $F : Q \to [0, 1]$ is the probability, for a state, to be final.*

We only consider here deterministic PFSA (called PDFA). That means that, given a state q and a symbol s, the state reached from the state q by the symbol s is unique if it exists. In order to define a probability distribution on Σ^* (the set of all strings built on Σ), each state of Q has to be able to generate a string with a strictly positive probability and, p and F must satisfy the following consistency constraint: $\forall q \in Q, F(q) + \sum_{a \in \Sigma} p(q, a) = 1$.

A string $s_0 \dots s_{n-1}$ is accepted by an automaton A iff there exists a sequence of states $e_0 \dots e_n$ such that: (i) $e_0 = q_0$, (ii) $\forall i \in [0, l-1], \delta(e_i, s_i) = e_{i+1}$, (iii) $F(e_n) \neq 0$. Then the automaton assigns to the string the following probability: $P_A(s_0 \dots s_{n-1}) = \left(\Pi_{i=0}^{n-1} p(e_i, s_i)\right) F(e_n)$. For example, the automaton in Figure 1, where states are represented by circles and final states by double circles, recognizes the string $baaa$ with probability $0.75 \times 1.0 \times 0.2 \times 0.2 \times 0.6 = 0.018$.

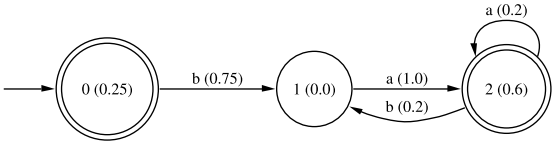


Figure 1: A PDFA with $q_0 = 0$ and its probabilities.

A lot of algorithms have been proposed to infer PDFA (Carrasco & Oncina 1994; Thollard, Dupont, & de la Higuera 2000; Kermorvant & Dupont 2002; Ron, Singer, & Tishby 1995). Most of them follow the same scheme based on state merging, summarized in Algorithm 1. Given a set of positive examples S_+, it first builds the *PPTA* (*Probabilistic Prefix Tree Acceptor*), which is an automaton accepting all the examples of S_+ (see part at the top of Figure 2). It is constructed such that the states corresponding to common prefixes are merged and each state and each transition are associated with the number of times they are used while parsing S_+. This number is then used to define the probability function p. If $C(q)$ is the number of times a state q is used while parsing S_+ and $C(q, a)$ is the number of times the transition (q, a) is used while parsing S_+, then $p(q, a) = \frac{C(q,a)}{C(q)}$. Similarly, if $C_f(q)$ is the number of times q is used as final state in S_+ for each state q, we have $F(q) = \frac{C_f(q)}{C(q)}$.

Data: S_+ training examples, **Result**: A a PDFA
$A \leftarrow$ build_PPTA(S_+)
while $(q_i, q_j) \leftarrow choose\_states(A)$ **do**
  if $compatible(q_i, q_j)$ **then** $merge(A, q_i, q_j)$
end
return A

Algorithm 1: Generic algorithm for inferring PDFA.

The second step of the algorithm consists in running through the $PPTA$ (function $choose\_states(A)$), and testing whether the considered states are statistically *compatible* (function $compatible(q_i, q_j)$). Several consecutive merging operations are done to keep the automaton structurally deterministic. The algorithm stops when no more merging is possible. For example, the automaton at the bottom part of Figure 2 represents the merging of the states labeled b and bab, and the ones labeled ba, baa, $baba$.

Theoretical Analysis in the Presence of Noise

Let us recall that our goal is to correct the *PPTA* constructed from noisy data with the help of a manually corrected sample. In this section, we study the impact of noise on the *PPTA* transition probabilities. We also show that the alphabet size has a direct influence on these probabilities. We assume that the noise is uniformly distributed on the letters of the learning sample, which corresponds to a reasonable hypothesis in front of the different possible types of errors

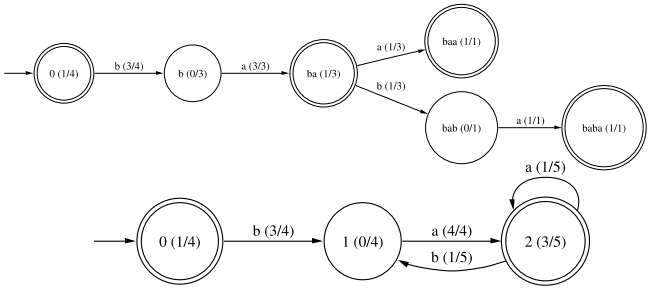


Figure 2: PPTA of $S_+ = \{ba, baa, baba, \lambda\}$ at the top (λ denotes the empty string). Below the PDFA obtained after two state merging.

(typing errors, experimental errors, *etc*). We denote by γ the noise level, corresponding to the percentage of corrupted letters. We assume that each noisy letter is replaced by a different one, randomly chosen in Σ. Let us consider a transition probability labeled by a letter $a \in \Sigma$ from a state q of the *PPTA*. We denote this probability by $p(q, a)$ in the absence of noise and by $p_\gamma(q, a)$ in the presence of noise.

Given a uniformly distributed noise γ, we can easily determine, after corruption, the transition probability of the *PPTA* relatively to γ, $p(q, a)$ and $|\Sigma|$:

$$\begin{aligned} p_\gamma(q,a) &= (1-\gamma)p(q,a) \\ &\quad +(1-F(q)-p(q,a))\frac{\gamma}{(|\Sigma|-1)} \quad (1) \\ &= (1-F(q)-|\Sigma|p(q,a))\frac{\gamma}{|\Sigma|-1} + p(q,a) \quad (2) \\ &= \left(1-\frac{|\Sigma|\gamma}{|\Sigma|-1}\right)p(q,a) + \frac{(1-F(q))\gamma}{(|\Sigma|-1)} \quad (3) \end{aligned}$$

Considering Equation 1, $p_\gamma(q, a)$ is composed of two terms. The first one represents the weighted proportion of letters a that are not affected by the noise, while the second corresponds to the weighted proportion of letters that were originally different from a before their modification into a (due to the noise). The main remark that we can state here, is that $p_\gamma(q, a)$ evolves linearly relatively to $p(q, a)$. This remark will justify, in the following section, our correction approach based on linear regression.

Let us now study the behavior of $p_\gamma(q, a)$ in function of γ.

Theorem 1 *Considering a uniformly distributed noise, $p_\gamma(q, a)$ is an increasing function of γ if $p(q, a) < \frac{1-F(q)}{|\Sigma|}$ and a decreasing function of γ otherwise.*

Proof. According to Equation 2,
$\frac{\partial p_\gamma(q,a)}{\partial \gamma} = (1 - F(q) - |\Sigma|p(q, a))\frac{1}{|\Sigma|-1}$.
Thus, $\frac{\partial p_\gamma(q,a)}{\partial \gamma}$ is positive when $p(q, a) < \frac{1-F(q)}{|\Sigma|}$ and negative otherwise. □

This result shows that when the noise level increases, the probability of a letter tends to a weighted value of $\frac{1-F(q)}{|\Sigma|}$. The alphabet size plays, then, an important role on a noisy distribution.

We show, now, that for a given level of noise, an increase in the number of letters tends to highly disturb the original distribution. For this purpose, we study the deviation between $p(q, a)$ and $p_\gamma(q, a)$.

Theorem 2 *$\Delta_\Sigma(q, a) = p(q, a) - p_\gamma(q, a)$ is an increasing function in $|\Sigma|$.*

Proof. Using Equation 1, we have:

$$\begin{aligned} \Delta_\Sigma(q,a) &= p(q,a) - (1-\gamma)p(q,a) \\ &\quad -(1-F(q)-p(q,a))\frac{\gamma}{(|\Sigma|-1)} \\ &= \gamma p(q,a) - (1-F(q)-p(q,a))\frac{\gamma}{(|\Sigma|-1)} \end{aligned}$$

Thus,
$\frac{\partial \Delta_\Sigma(q,a)}{\partial \Sigma} = -\frac{-(1-F(q)-p(q,a))\gamma}{(|\Sigma|-1)^2} = \frac{(1-F(q)-p(q,a))\gamma}{(|\Sigma|-1)^2} \geq 0$ □

The deviation $\Delta_\Sigma(q, a)$ increases with $|\Sigma|$ and implies that the probability distribution is highly modified, whatever the level γ. This result can be easily interpreted in the presence of noise: given a large alphabet, a part of letters a are corrupted, and in parallel, only a small proportion of letters, originally different from a, are changed in a. Thus, whatever γ, $p(q, a)$ is modified. This important remark shows that a noise correction procedure must consider both γ and $|\Sigma|$.

Correcting Uniformly Noisy Samples

Probabilistic automata learning is based on the frequencies of transitions and states. The presence of noise can highly disturb the estimation of some transitions and have a negative impact on the quality of the inferred automata. To tackle this drawback, we propose to correct the transition probabilities of the *PPTA* with the help of a small manually corrected sample. We already argued about the interest of such an approach. An important remark can be made here. While the corrected sample will help us to estimate the noise level, this sample can not, in any case, be used as a learning sample to infer a PDFA. Indeed, statistically speaking, this sample would be too small to estimate accurately the whole set of transitions, which would represent a too high number of parameters. The estimation of the single parameter γ is, on the other hand, more reasonable. In this context, we begin to show how we exploit the corrected sample to estimate γ, before presenting how we use it to correct the *PPTA*.

Noise Estimation by Linear Regression

Let us consider two *PPTA*: $A_b = (Q_b, \Sigma, \delta_b, q_0^b, F_b)$ built from the noisy sample, and $A_{\overline{b}} = (Q_{\overline{b}}, \Sigma, \delta_{\overline{b}}, q_0^{\overline{b}}, F_{\overline{b}})$ from the manually corrected one. Let $q_b \in Q_b$ a state of A_b, $w \in \Sigma^*$ such that $\delta_b(q_0^b, w) = q_b$ and $q_{\overline{b}} \in Q_{\overline{b}}$ such that $\delta_{\overline{b}}(q_0^{\overline{b}}, w) = q_{\overline{b}}$. $q_{\overline{b}}$ is the state recognizing the same prefix as q_b. We assume that $C(q_{\overline{b}}) = n_{\overline{b}}$ and $C(q_b) = n_b$. We consider the following set of pairs of transition probabilities: $E_{init} = \{(p_{\overline{b}}(q_{\overline{b}}, a), p_b(q_b, a))|a \in \Sigma\}$.

Each pair of reals defines a point in $[0; 1]^2$. According to Equation 3, there is a linear dependence between the values of the probabilities of noisy transitions and noise-free ones, defined by an equation of the form $y = C * x + d$.
C, the line slope, and d are defined by (Cox 1987): $C = \frac{\sum_{a\in\Sigma}(p_b(q_b,a)-\frac{1}{|\Sigma|})*(p_{\overline{b}}(q_{\overline{b}},a)-\frac{1}{|\Sigma|})}{\sum_{a\in\Sigma}(p_{\overline{b}}(q_{\overline{b}},a)-\frac{1}{|\Sigma|})^2}$, and $d = \frac{1}{|\Sigma|} - C * \frac{1}{|\Sigma|}$.

Using C and Equation 3, we estimate γ such that:

$$C = \left(1 - \frac{|\Sigma|\gamma}{|\Sigma|-1}\right) \Rightarrow \gamma = \frac{(|\Sigma|-1)*(1-C)}{|\Sigma|} \quad (4)$$

However, probability estimations of $A_{\overline{b}}$ are accurate relatively to the size of the corrected sample. Rather than using directly these estimations, we propose to compute the bounds of a confidence interval which takes into account the size of $A_{\overline{b}}$. Given an estimation $\hat{p}_{\overline{b}}(q_{\overline{b}}, a)$ of a probability $p_{\overline{b}}(q_{\overline{b}}, a)$ estimated from $n_{\overline{b}}$ instances, a confidence interval, with a risk α, around $p_{\overline{b}}(q_{\overline{b}}, a)$ is defined by:

$$\left[\hat{p}_{\overline{b}}(q_{\overline{b}}, a) - u_{\alpha/2}\sqrt{\frac{\hat{p}_{\overline{b}}(q_{\overline{b}},a)\overline{\hat{p}_{\overline{b}}(q_{\overline{b}},a)}}{n_{\overline{b}}}}\ ;\ \hat{p}_{\overline{b}}(q_{\overline{b}}, a) + u_{\alpha/2}\sqrt{\frac{\hat{p}_{\overline{b}}(q_{\overline{b}},a)\overline{\hat{p}_{\overline{b}}(q_{\overline{b}},a)}}{n_{\overline{b}}}}\right]$$

where $\overline{\hat{p}_{\overline{b}}(q_{\overline{b}}, a)} = 1 - \hat{p}_{\overline{b}}(q_{\overline{b}}, a)$, and $u_{\alpha/2}$ corresponds to the $(1-\alpha/2)$ percentile of the normal distribution.

We compute this interval for each couple of transition probabilities. We obtain two sets, E_{inf} corresponding to the pairs of transitions of the lower bounds of each interval, and E_{sup} to the upper ones:
$E_{inf} = \{(\hat{p}_{\overline{b}}(q_{\overline{b}}, a)^-, \hat{p}_b(q_b, a)^-) | a \in \Sigma\}$,
$E_{sup} = \{(\hat{p}_{\overline{b}}(q_{\overline{b}}, a)^+, \hat{p}_b(q_b, a)^+) | a \in \Sigma\}$.
Where $\hat{p}_x(q_x, a)^- = \hat{p}_x(q_x, a) - u_{\alpha/2}\sqrt{\frac{\hat{p}_x(q_x,a)(1-\hat{p}_x(q_x,a))}{n_x}}$
and $\hat{p}_x(q_x, a)^+ = \hat{p}_x(q_x, a) + u_{\alpha/2}\sqrt{\frac{\hat{p}_x(q_x,a)(1-\hat{p}_x(q_x,a))}{n_x}}$.

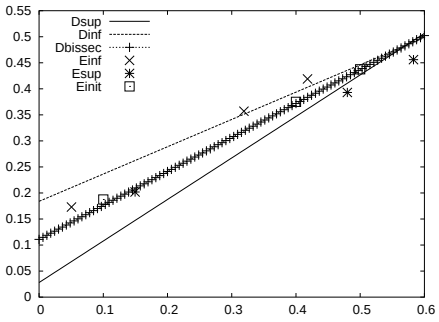
We use each of the sets of points in order to obtain two lines of regression $y = C_{sup} * x + d_{sup}$ and $y = C_{inf} * x + d_{inf}$ defining the minimal and maximal values for γ that can be obtained, taking into account the sizes of both of the noisy and corrected samples. Finally, we estimate γ, according to Equation 4, with the slope of the bisecting of these two lines, defined by the equation:

$$\frac{C_{sup}*x - y + d_{sup}}{\sqrt{C_{sup}^2+1}} = \frac{y - C_{inf}*x - d_{inf}}{\sqrt{C_{inf}^2+1}}$$

Let us consider an example to illustrate our approach. Let $\Sigma = \{a, b, c\}$ and $\gamma = 25\%$. Figure 3(a) indicates the transition probabilities for the two states $q_{\overline{b}}$ and q_b respectively estimated from 100 and 1000 examples. Then, we compute, using a risk $\alpha = 10\%$, the sets $E_{sup} = \{(0.1494, 0.2019), (0.48, 0.393), (0.583, 0.456)\}$ and $E_{inf} = \{(0.0505, 0.173), (0.319, 0.357), (0.418, 0.149)\}$. Using these two samples, we compute the two regression lines, respectively $Dsup$ and $Dinf$, and their bisecting line $Dbissec$. The result is presented on Figure 3(b). The slope of the bisecting line is 0.652376 and provides an estimation $\hat{\gamma} = 23.2\%$. This estimation is relatively closed to the theoretical value γ, despite the fact that less than 10% of the examples have been corrected.

$l \in \Sigma$	$\hat{p}_{\overline{b}}(q_{\overline{b}}, l)$	$\hat{p}_b(q_b, l)$
a	0.1	0.1875
b	0.4	0.3750
c	0.5	0.4375

(a) Probabilities of $q_{\overline{b}}$ and q_b.



(b) Regression lines.

Figure 3: An example of noise estimation.

Correcting the *PPTA*

According to Equation 2, a noisy transition probability evolves linearly with γ. After estimating this level of noise, we can correct the probability $p_\gamma(q_b, a)$ by:

$$p_\gamma(q_b, a) = \frac{(|\Sigma|-1)p_\gamma(q_b,a) - (1-F(q_b))\gamma}{(|\Sigma|-1)-\gamma|\Sigma|}$$

More precisely, rather than correcting probabilities, we correct the frequencies:

$$C(q_b, a) = \frac{(|\Sigma|-1)C(q_b,a) - (C(q_b) - C_f(q_b))\gamma}{(|\Sigma|-1)-\gamma|\Sigma|}$$

Note that a given correction has an effect on the states accessible from the corrected transitions. We need then to take into account these changes in the *PPTA*. We use a breadth first search of the *PPTA* to guarantee an efficient correction. Formally, for each state q that recognizes a string coming from states directly accessible from q_b, the outgoing transitions of q are modified as follows. $\forall w \in \Sigma^*$, $\forall a \in \Sigma$ such that $\exists q \in Q_b$ with $\delta'(\delta(q_b, a), w) = q$, $\forall e \in \Sigma \cup \{\#\}$:

$$C(\delta'(\delta(q_b, a), w), e) = \frac{(|\Sigma|-1)*C(\delta'(\delta(q_b,a),w),e) - \gamma * N_{w,e}}{|\Sigma|-1-\gamma*|\Sigma|}$$

where $N_{w,e} = \sum_{c \in \Sigma} C(\delta'(\delta(q_b, c), w), e)$ represents the amount of letters e, accessible from the string w, that can be affected by the correction of q_b, and $C(q, \#) = C_f(q)$. We summarize our approach in Algorithm 2.

Name: $correct\_state$
Data: A_b a noisy *PPTA*, $A_{\overline{b}}$ a corrected *PPTA*, q_b a state of A_b
begin
  γ = Estimate of the level of noise in q_b using $A_{\overline{b}}$
  foreach $a \in \Sigma$ **do**
    $C(q_b, a) \leftarrow \frac{(|\Sigma|-1)*C(q_b,a) - (C(q_b) - C_f(q_b))*\gamma}{(|\Sigma|-1)-\gamma*|\Sigma|}$
    //Correction of $C(q_b, a)$
    $C(\delta(q_b, a)) \leftarrow C(q_b, a)$ //Correction step
  end
  Adapt the frequencies of transitions and states accessible from q_b
  foreach $a \in \Sigma$ **do** $correct\_state(A_b, A_{\overline{b}}, \delta(q_b, a))$
end

Algorithm 2: Algorithm for correcting probabilities.

Experimental Evaluation

In this section, we present a set of experiments allowing us to evaluate the relevance of our approach in the presence of noisy data. For this purpose, we use the algorithm Alergia (Carrasco & Oncina 1994), which is one of the best known state merging algorithms. We compare its performance with and without a *PPTA* correction, according to two criteria. The first one corresponds to the case where the target automaton is *a priori* known. In this case, we measure a distance between the inferred model and the target to evaluate the learned model quality. The second one corresponds to the case where the target automaton is unknown. In this context, we evaluate the inferred models with a perplexity measure on a test set.

Evaluation Criteria

Probabilistic Distance Lyngsø *et al.* defined distances between two Hidden Markov Models introducing the co-emission probability, as the probability that two independent models generate the same string (Lyngsø, Pedersen, & Nielsen 1999). (Carrasco & Rico-Juan 2002) presents an adaptation of the co-emission to stochastic tree automata. The probability that two probabilistic models A and A' generate the same strings is defined by: $C(A, A') = \sum_{s \in \Sigma^*} P_A(s) P_{A'}(s)$, where $P_A(s)$ is the probability of s given the model A. This co-emission probability allows us to define a probabilistic distance D_a, which can be interpreted as the measure of the angle between two vectors representing probabilistic automata in a space where base is the set Σ^*:

$$D_a(A, A') = \arccos\left(\frac{C(A,A')}{\sqrt{C(A,A)C(A',A')}}\right)$$

Perplexity The quality of a probabilistic automaton A on a set of sequences S, can be evaluated by computing the mean of the log-likelihood of the elements of S relatively to the distribution defined by A: $LL = (\frac{1}{\|S\|}\sum_{j=1}^{|S|} \log p(s_j \mid A))$, where $p(s_j \mid A)$ is the probability of the j^{th} string of S. A perfect model can predict each element of the test set with a probability equal to one, and so $LL = 0$. In a general way, we consider the perplexity of the test set which is defined by $PP = 2^{LL}$. A minimal perplexity ($PP = 1$) is reached when the model can predict each element of the test sample. Therefore we consider that a model is more predictive than another if its perplexity is lower.

Experimentations

We carried out two series of experiments: one when the target is unknown and one when it is *a priori* known. For the first series, we used 3 datasets of the UCI Irvine (Blake & Merz 1998): AGARICUS, TICTACTOE and BADGES containing respectively 8126, 769 and 530 examples. Each base is composed of a positive and a negative set, used for classification tasks. We decided to use the negative sample as another concept to learn, which allows us to have 3 additional datasets. We also used a database of world first names beginning with the letter A (FIRSTNAME). This base has 1045 male first names (M) and 842 female ones (F). For the series with a known target model, we used three sources.

$Base$	M	WN	N	C	U	PA
AGARICUS+	P_e	1.6	4.1±2.9	3.3±0.2	4.1±2.9	3.1±0.9
AGARICUS-	P_e	1.7	4.1±2.9	2.8±0.1	3.5±1.4	2.9±0.8
BADGES+	P_e	25	29±4.1	28±0.6	31±2.8	28±4.1
BADGES-	P_e	26	32±4.3	31±0.1	32±2.9	30±3.9
TICTACTOE+	P_e	2.8	3.6±0.7	3.3±0.2	3.4±0.1	3.7±0.5
TICTACTOE-	P_e	3.0	3.6±0.6	3.2±0.1	3.4±0.1	3.6±0.5
FIRSTNAME M	P_e	11	15±2.9	11±0.1	16±4.1	13±2.7
FIRSTNAME F	P_e	12	17±3.2	11±0.5	14±3.1	14±3.0
ART1	P_e	1.7	2.0±0.3	1.8±0.1	1.7±0.1	2.0±0.3
ART2	P_e	1.3	1.7±0.4	1.3±0.1	1.4±0.1	1.6±0.3
REBER1	P_e	1.6	2.6±1.1	1.8±0.2	2.4±0.8	2.3±0.8
REBER2	P_e	1.6	2.1±0.6	1.8±0.1	1.6±0.1	2.1±0.5
REBER3	P_e	1.6	2.1±0.5	1.9±0.1	1.6±0.1	2.1±0.5
Average	P_e	**7.0**	**9.2±1.9**	**7.9±0.2**	**8.9±1.4**	**8.3±1.5**
ART1	D_a	0.01	0.4±0.3	0.2±0.1	0.8	0.5±0.4
ART2	D_a	0.36	0.7±0.3	0.4±0.2	0.8	0.8±0.4
REBER1	D_a	0.07	1.0±0.5	0.5±0.2	0.5	0.5±0.4
REBER2	D_a	0.01	0.4±0.4	0.2±0.1	0.1	0.3±0.3
REBER3	D_a	0.01	0.4±0.4	0.3±0.1	0.1	0.4±0.3
Average	D_a	**0.11**	**0.6±0.4**	**0.3±0.1**	**0.5**	**0.5±0.4**

Table 1: Results obtained on 18 datasets.

The *Reber* grammar (Reber 1967), from which we generated three samples: REBER1 with 500 examples, REBER2 with 5000 examples and REBER3 composed of 10000 examples. We also used two target automata recognizing the languages $ac^*a + bc^*b$ (ART1) and $(aa)^*(bbb)^*$ (ART2). We generated a sample of 10000 instances for ART1 and 2000 for ART2.

Our experimental set up consisted in adding noise to each learning sample, with different levels, from 1% to 50%. We then randomly chose a given proportion of examples to construct the sample manually corrected by the expert. This sample represents 25% of the original one for datasets having less than 400 examples, 10% for those having less than 2000 examples and 5% for the others, except for those with 10000 examples where it represents 2.5% of the original size. We applied our correction method, for each corrupted file, using a risk of 5% for the computation of confidence intervals. Finally, for each obtained file (noise-free, noisy, noisy and corrected), we learned an automaton using the algorithm Alergia. The quality criterion is D_a for samples where the target automaton is known and the perplexity for the others. For this last measure, we used a 5 fold cross-validation. Moreover, in order to compare our ability to deal with noisy data, we compared our approach with the results obtained by the adaptation of Alergia proposed in (Habrard, Bernard, & Sebban 2003a) (denoted by PA) on noisy samples. This adaptation allows one to reduce the impact of noise using a more restrictive merging rule.

The results are presented in Table 1. The name of the samples is indicated in column $Base$ and the quality measure (D_a for the distance or P_e for the perplexity) in column M. The optimal result, the one obtained with a **whole** noise-free sample, is indicated in column WN. Column N describes results obtained from the uncorrected files, column C those after our automatic correcting method, and column PA, those obtained with the adaption PA. For these columns, we indicate the average of the results on all the levels of noise ± the standard deviation. In order to verify that inferring the automaton only from the small file corrected by the expert is not sufficient, we present in column U the results obtained in this context.

For all the databases, automata inferred from samples corrected with our method (C) are significantly better than

those learned directly from the noisy samples (N). Moreover, our technique is better, on average, than the approach PA which confirms its interest and relevance in the context of dealing with noise in probabilistic grammatical inference. We can also note that the standard deviation is significantly smaller, that denotes a good robustness of the approach. Moreover, our results confirm that the use of the small manually corrected sample alone (column U) is not sufficient to learn a general performing model. Indeed, automata inferred after our automatic correcting phase are, on average, better than those learned only from the manually corrected sample. An important point concerns the size of the manually corrected sample. On Figure 4, we represent the perplexity evolution on the base REBER1, relatively to the corrected sample size, for C and U and a level of noise of 25%. The automaton, obtained after correction, is better until a size representing less than 20% of the original sample, that corresponds to less than 100 examples. Beyond this point, the manually corrected sample is logically self-sufficient.

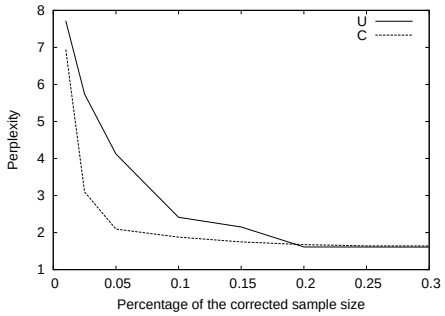


Figure 4: Impact of the corrected sample size.

Conclusion

In this paper, we proposed a technique allowing us to automatically correct a learning sample containing a uniformly distributed noise. Our approach is doubly interesting: firstly, we are able to correct the learning sample given the level of noise. Secondly, we provide a solution for estimating this level with the help of a small sample corrected by an expert. We experimentally showed the interest of our method in the context of learning probabilistic automaton from noisy data.

We aim at extending our approach to process other kinds of noise (such as Gaussian noise for example). While the estimation strategy of γ would not be changed, we have actually to define another functional relationship between values of the transition probabilities of the corrupted *PPTA* and those of the manually corrected one.

Another remark concerns the size of the corrected sample. When it is high, the correction is obviously efficient, from a statistical point of view. However, the correction of too large a sample, by a human expert, is not reasonable. Then, we would like to theoretically study the minimal size for an efficient correction.

References

Blake, C., and Merz, C. 1998. University of California Irvine repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/`.

Brodley, C., and Friedl, M. 1996. Identifying and eliminating mislabeled training instances. In *13th National Conference on Artificial Intelligence AAAI/IAAI*, 799–805.

Carrasco, R., and Oncina, J. 1994. Learning stochastic regular grammars by means of a state merging method. In *Grammatical Inference and Applications, ICGI'94*, number 862 in LNAI, 139–150. Springer Verlag.

Carrasco, R. C., and Rico-Juan, J. R. 2002. A similarity between probabilistic tree languages: application to xml document families. *Pattern Recognition* 36(9):2197–2199.

Cox, C. P. 1987. *A Handbook of Introductory Statistical Methods*. Wiley & Sons.

de la Higuera, C. 2005. A bibliographical study of grammatical inference. *Pattern Recognition*. (to appear).

Habrard, A.; Bernard, M.; and Sebban, M. 2003a. Improvement of the state merging rule on noisy data in probabilistic grammatical inference. In *14th European Conference on Machine Learning*, volume 2837 of *LNAI*, 169–180. Springer.

Habrard, A.; Bernard, M.; and Sebban, M. 2003b. Probabilistic approach for reduction of irrelevant tree-structured data. In *1st International Workshop on Mining Graphs, Trees and Sequences (MGTS-2003)*, 11–20.

John, G.; Kohavi, R.; and Pfleger, K. 1994. Irrelevant features and the subset selection problem. In *11th International Conference on Machine Learning*, 121–129.

Kermorvant, C., and Dupont, P. 2002. Stochastic grammatical inference with multinomial tests. In *6th International Colloquium on Grammatical Inference (ICGI 2000)*, volume 2484 of *LNAI*, 149–160. Amsterdam: Springer.

Lyngsø, R.; Pedersen, C.; and Nielsen, H. 1999. Metrics and similarity measures for hidden Markov models. In *7th International Conference on Intelligent Systems for Molecular Biology (ISMB '99)*, 178–186. AAAI Press.

Reber, A. 1967. Implicit learning of artificial grammars. *Journal of verbal learning and verbal behaviour* 6:855–863.

Ron, D.; Singer, Y.; and Tishby, N. 1995. On the learnability and usage of acyclic probabilistic automata. In *Computational Learning Theory, COLT'95*, 31–40.

Sebban, M., and Janodet, C. 2003. On state merging in grammatical inference: A statistical approach for dealing with noisy data. In *20th International Conference on Machine Learning (ICML 2003)*, 688–695.

Sebban, M., and Nock, R. 2000. Instance pruning as an information preserving problem. In *17th International Conference on Machine Learning (ICML)*, 855–862.

Thollard, F.; Dupont, P.; and de la Higuera, C. 2000. Probabilistic dfa inference using kullback–leibler divergence and minimality. In *17th International Conference on Machine Learning*, 975–982. Morgan Kauffman.